What is claimed is:

1. A computer system for executing predicated instructions wherein each instruction includes a guard, the value of which determines whether or not that instruction is executed, the computer system comprising:

a fetch unit for fetching instructions to be executed;

a decode unit for decoding said instructions;

at least one pipelined execution unit for executing decoded instructions and being associated with a guard register file holding values of the guards to allow resolution of the guards to be made to determine whether an instruction is committed; and

an emulation unit including control circuitry which cooperates with the decode unit to selectively control the decode unit to implement step-by-step execution of an instruction sequence wherein, for each committed instruction, a divert routine is executed by the computer system and for each non-committed instruction the next instruction in the instruction sequence is executed.

2. A computer system according to claim 1, which is implemented on a single chip.

3. A computer system according to claim 1, which includes a program memory for holding said instructions to be executed.

4. A computer system according to claim 1, wherein the emulation unit is associated with an emulation program memory which holds a plurality of divert routines.

5. A computer system according to claim 1, wherein, for each instruction in the sequence, the decode unit is operable to issue a request to the execution pipeline for guard resolution, the guard resolution being transmitted to the control circuitry of the emulation unit which implements said divert routine if the instruction is committed.

6. A computer system according to claim 1, wherein the divert routine comprises a sequence of debug instructions, each debug instruction being associated with at least one debug attribute.

7. A computer system according to claim 6, wherein the last instruction in the divert routine includes a stall attribute which places the decode unit in a stall state.

8. A computer system according to claim 6, wherein the last instruction in the divert routine includes an atomic attribute which inhibits execution of any instruction other than the next instruction in the step-by-step sequence.

9. A computer system according to claim 6, wherein the last instruction in the divert routine restores the interrupted instruction sequence.

10. A computer system according to claim 1, which is connected to a host computer which can take over operation of the emulation unit responsive to certain debug conditions.

11. A computer system according to claim 1, which includes a microinstruction generator which receives instructions from the decode unit and supplies microinstructions to the execution pipeline, said microinstructions including fields for holding respective guards to be resolved.

12. A computer system according to claim 1, which includes a plurality of parallel pipelined execution units, including at least two data unit pipelines for executing data processing instructions and at least two address unit pipelines for executing memory access instructions.

13. A method of executing predicated instructions wherein each instruction includes a guard, the value of which determines whether or not that instruction is executed, the method comprising:

fetching each of a sequence of instructions to be executed;

decoding each instruction and requesting resolution of its guard to determine whether the instruction is committed; and

if the instruction is committed, implementing a divert routine whereby debug code is executed and, if the instruction is not committed, fetching and decoding the next instruction in the instruction sequence.

14. A method according to claim 13, wherein guard resolution is effected in a pipelined execution unit for executing said instructions.

15. A method according to claim 13, wherein the debug code includes a last instruction which has a stall attribute which implements a stall state at the end of the debug code.

16. A method according to claim 13, wherein the debug code includes a last instruction which has an atomic attribute which inhibits execution of any instruction other than the next instruction in the step by step sequence.

17. A computer system for executing instructions in a first, user mode and a second, debug mode, the computer system comprising:

a first store for holding user instructions;

a second store for holding debug instructions, wherein the debug instructions are held in the second store in association with debug attributes, wherein said debug attributes include a stall attribute;

a fetch unit for selectively fetching instructions from the first or second store depending on the mode of the computer system;

a decode unit for decoding said instructions and reading said attributes; and

a emulation unit which includes control circuitry which cooperates with the decode unit to selectively set the decode unit into a stall state by issuance of a stall signal;

wherein the decode unit includes stall control circuitry which is responsive to reading of a stall attribute or receipt of a stall signal from the emulation unit to place the decode unit into a stall state.

18. A computer system according to claim 17, which is implemented on a single chip.

19. A computer system according to claim 17, wherein the debug code comprises a plurality of divert routines allowing debug functions to be implemented by the computer system.

20. A computer system according to claim 17, wherein the debug attributes include an atomic attribute which inhibits execution of any instruction other than the next instruction in the sequence of instructions being executed.

21. A computer system according to claim 17, which is connected to a host computer which can take over operation of the emulation unit responsive to certain debug conditions.

22. A computer system according to claim 17, which includes at least one pipelined execution unit for executing said instructions.

23. A computer system according to claim 22 for executing predicated instructions, wherein each instruction includes a guard, the value of which determines whether or not that instruction is executed, said at least one pipelined execution unit being associated with a guard register file holding values of the guards to allow resolution of the guards to be made to determine whether an instruction is committed.

24. A method of setting a stall state of a computer system which comprises a fetch unit for fetching instructions to be executed and a decode unit for decoding said instructions, wherein the stall state is set selectively at the decode unit by reading

stall attributes associated with debug instructions in a debug mode, or by receipt of a stall command responsive to certain conditions when executing user instructions in a user mode.